Introduction & Vector Representations of Text

COM4513/6513 Natural Language Processing

Nikos Aletras
n.aletras@sheffield.ac.uk

@nikaletras

Computer Science Department

Week 1
Spring 2020

The University Of Sheffield.

Part I: Introduction

Course Practicalities

Lecture slides & lab assignments:

- https://sheffieldnlp.github.io/com4513-6513/

Lab demonstrators:

- George Chrysostomou
- Hardy
- Katerina Margatina
- Danae Sanchez Villegas
- Peter Vickers
- Zeerak Waseem

Course Practicalities

Google Group (join using `@sheffield.ac.uk`)

- `https://groups.google.com/a/sheffield.ac.uk/forum/?hl=en-GB#!forum/com4513-6513---nlp-2020-group`

Office hours:

- **Thursdays 13:10-14:00**, Regent Court, G36b (First come, first served)

Assessment

- **60% exam** where **everything** is assessed: lecture slides, bibliographical references, classroom discussion, lab content, etc.
- **40% 2 lab assignments** (20% each):
  - Deadlines TBA
  - Do them (so that we can help) and do not plagiarise!

Feedback

To you:

- During the lab sessions on assignments
- Questions in class and the Google group
- During office hours

Feedback

To you:

- During the lab sessions on assignments
- Questions in class and the Google group
- During office hours

And to me:

- NSS evaluation
- Module evaluation

Feedback

To you:

- During the lab sessions on assignments
- Questions in class and the Google group
- During office hours

And to me:

- NSS evaluation
- Module evaluation

- Some changes from last year were student suggestions.

Course goals

- Learn how to develop systems to perform natural language processing (NLP) tasks
- Understand the main machine learning (ML) algorithms for learning such systems from data
- Become familiar with important NLP applications
- Have knowledge of state-of-the art NLP and ML methods

Prerequisites

Essential

- Text Processing (COM3110/4115/6115)
- Programming skills (i.e. Python3) see the Python Introduction for NLP.

Prerequisites

Essential

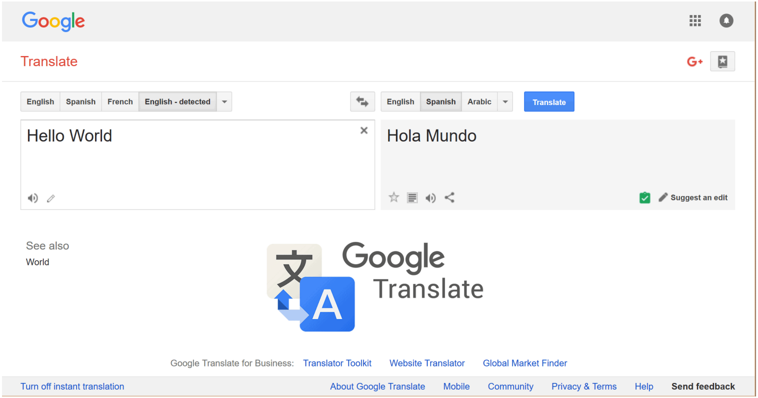
- Text Processing (COM3110/4115/6115)
- Programming skills (i.e. Python3) see the Python Introduction for NLP.

Optional (but strongly recommended)

- Machine Learning and Adaptive Intelligence (COM4509/6509).
- Basic Linguistics: see this tutorial by Emily Bender

Why Natural Language Processing?

Why Natural Language Processing?



Machine Translation

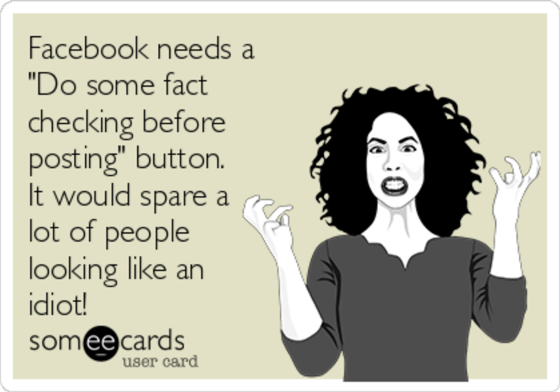
Why Natural Language Processing?

Question Answering

Why Natural Language Processing?

Playing Jeopardy

Why Natural Language Processing?



Fact Checking

Why Natural Language Processing?

Support The Guardian

Contribute → Subscribe →

Sign in

The Guardian

News Opinion Sport Culture Lifestyle

UK World Business Football UK politics Environment Education Society Science More

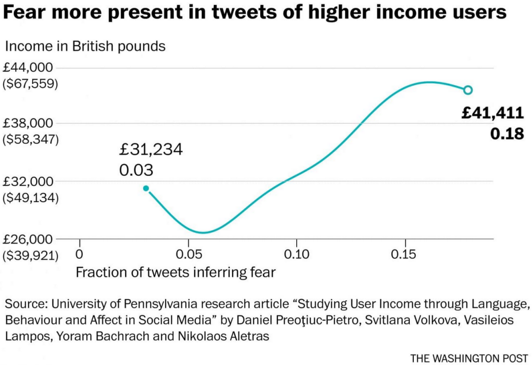
Artificial intelligence (AI)

Artificial intelligence 'judge' developed by UCL computer scientists

Software program can weigh up legal evidence and moral questions of right and wrong to predict the outcome of trials

Assist in Human Decision Making

Why Natural Language Processing?



Computational Social Science

Why is NLP challenging?

Why is NLP challenging?

- Natural languages (unlike programming languages) are not designed; they **evolve**!
  - new words appear constantly
  - parsing rules are flexible
  - ambiguity is inherent

Why is NLP challenging?

- Natural languages (unlike programming languages) are not designed; they **evolve**!
  - new words appear constantly
  - parsing rules are flexible
  - ambiguity is inherent
- world knowledge is necessary for interpretation

Why is NLP challenging?

- Natural languages (unlike programming languages) are not designed; they **evolve**!
  - new words appear constantly
  - parsing rules are flexible
  - ambiguity is inherent
- world knowledge is necessary for interpretation
- many languages, dialects, styles, etc.

Why statistical NLP?

- Traditional rule-based artificial intelligence (symbolic AI):
  - requires expert knowledge to engineer the rules
  - not flexible to adapt in multiple languages, domains, applications
- Learning from data (machine learning) adapts: Easier
  - to evolution: just learn from new data
  - to different applications: just learn with the appropriate target representation Accord with different situation

Words of caution

- When exploring a task, it is often useful to experiment with some simple rules to test our assumptions
- In fact, for some tasks rule-based approaches rule, especially in industry:
  - question answering
  - natural language generation

Words of caution

- When exploring a task, it is often useful to experiment with some simple rules to test our assumptions
- In fact, for some tasks rule-based approaches rule, especially in industry:
  - question answering
  - natural language generation
- If we don't know how to perform a task, it is unlikely that a ML algorithm will find it out for us

NLP =? ML No

- NLP is a confluence of computer science, artificial intelligence (AI) and linguistics
- ML provides statistical techniques for problem solving by learning from data (current dominant AI paradigm)
- ML is **often** used in modelling NLP tasks

NLP =? Computational Linguistics

- Both mostly use text as data
- In Computational Linguistics (CL), computational/statistical methods are used to support the study of linguistic phenomena and theories
- In NLP, the scope is more general. Computational methods are used for translating text, extracting information, answering questions etc.

NLP =? Computational Linguistics

- Both mostly use text as data
- In Computational Linguistics (CL), computational/statistical methods are used to support the study of linguistic phenomena and theories
- In NLP, the scope is more general. Computational methods are used for translating text, extracting information, answering questions etc.

The top NLP scientific conference is called:
Annual Meeting of the Association for Computational Linguistics (ACL)

Other Related fields

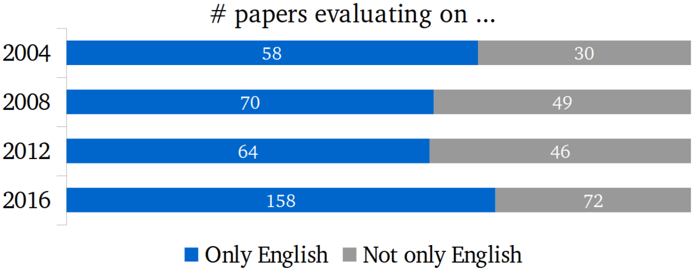
- Speech Processing
- Machine Learning
- Artificial Intelligence
- Search & Information Retrieval
- Statistics
- Any field that involves processing language:
  - literature, history, etc. (i.e. digital humanities)
  - biology
  - social sciences (sociology, psychology, law)

Some food for thought

- 6,000 languages in the world, but in research papers?

Some food for thought

- 6,000 languages in the world, but in research papers?



http://sjmielke.com/acl-language-diversity.htm

- NLP research has an English bias, our work is cut out!

Course overview

- **Lecture 1:** Introduction and Vector Representations of Text
- **Lecture 2:** Text Classification with Logistic Regression
- **Lecture 3:** Language Modelling with Hidden Markov Models
- **Lecture 4:** Part-of-Speech Tagging with Conditional Random Fields
- **Lecture 5:** Dependency Parsing

Course overview (cont.)

- **Lecture 6:** Information Extraction and Ethics Best Practices for NLP by Prof. Jochen Leidner
- **Lecture 7:** Feed-forward Networks and Neural Word Vectors
- **Lecture 8:** Recurrent Networks and Neural Language Modelling
- **Lecture 9:** Neural Seq2Seq Models for Machine Translation and Summarisation
- **Lecture 10:** Transfer Learning for NLP

Course Bibliography

- Jurafsky and Martin. 2008. Speech and Language Processing, Prentice Hall [3rd edition]
- Christopher D. Manning and Hinrich Schütze. 1999. Foundations of Statistical Natural Language Processing, MIT Press.
- Yoav Goldberg. 2017. Neural Network Methods in Natural Language Processing (Synthesis Lectures on Human Language Technologies), Morgan & Claypool Publishers, [A Primer on Neural Networks for NLP]
- Jacob Eisenstein. 2019. Introduction to Natural Language Processing. MIT Press. (A draft can be found here)
- other materials referenced at the end of each lecture

Opinion Poll time!

How long do you think it will take us to develop NLP systems that understand human language and have intelligence similar to humans?

Cast your vote here: `https://tinyurl.com/vgmtwvd`

Part II: Vector Representations of Text

- Why do we need vector representations of text?
- How can we transform a raw text to a vector?

Vectors and Vector Spaces

- A vector (i.e. embedding) $\mathbf{x}$ is a **one-dimensional array** of d elements (coordinates), that can be identified by an index $i \in d$. e.g. $x_1 = 0$

x	2	0	...	5
index	1	2	...	d

Vectors and Vector Spaces

- A vector (i.e. embedding) **x** is a **one-dimensional array** of d elements (coordinates), that can be identified by an index $i \in d$. e.g. $x_1 = 0$

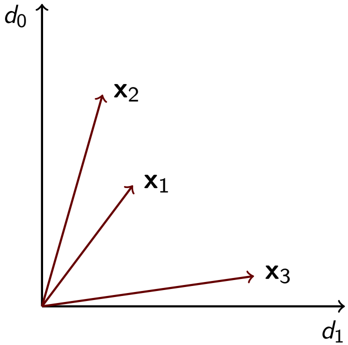
x	2	0	...	5
index	1	2	...	d

- A collection of n vectors is a **matrix** X with size $n \times d$ - also called a **vector space**. e.g. $X[2, 1] = -2$

$$n \left\{ \underbrace{\begin{array}{|c|c|c|c|} \hline 2 & 0 & \dots & 5 \\ \hline -2 & 9 & \dots & 0 \\ \hline \multicolumn{4}{|c|}{\dots} \\ \hline 0 & 2 & \dots & 0 \\ \hline \end{array}}_{d} \right.$$

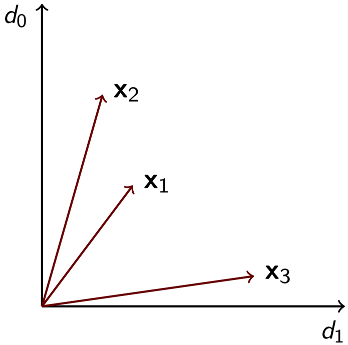
- Note that in Python indices start from 0!

Example of a vector space



d_i $(i \in 1, 2)$ are the coordinates and $\mathbf{x}_j$ are vectors

Example of a vector space



d_i $(i \in 1, 2)$ are the coordinates and $\mathbf{x}_j$ are vectors

How can we measure that $\mathbf{x}_1$ is closer to $\mathbf{x}_2$ than to $\mathbf{x}_3$?

Vector Similarity

- **Dot (inner) product:** takes two equal-length sequences of numbers (i.e. vectors) and returns a single number.

$$\mathrm{dot}(\mathbf{x_1}, \mathbf{x_2}) = \mathbf{x_1} \cdot \mathbf{x_2} = \mathbf{x_1}\mathbf{x_2}^\top = \sum_{i=1}^{d} x_{1,i}x_{2,i} = x_{1,1}x_{2,1} + \ldots + x_{1,d}x_{2,d}$$

Vector Similarity

- **Dot (inner) product:** takes two equal-length sequences of numbers (i.e. vectors) and returns a single number.

$$\text{dot}(\mathbf{x_1}, \mathbf{x_2}) = \mathbf{x_1} \cdot \mathbf{x_2} = \mathbf{x_1}\mathbf{x_2}^\top = \sum_{i=1}^{d} x_{1,i}x_{2,i} = x_{1,1}x_{2,1} + ... + x_{1,d}x_{2,d}$$

- **Cosine similarity:** normalise dot product ($[0, 1]$) by dividing with vectors' lengths (or magnitude or norm) $|\mathbf{x}|$.

$$\text{cosine}(\mathbf{x_1}, \mathbf{x_2}) = \frac{\mathbf{x_1} \cdot \mathbf{x_2}}{|\mathbf{x_1}||\mathbf{x_2}|} = \frac{\sum_{i=1}^{d} x_{1,i}x_{2,i}}{\sqrt{\sum_{i=1}^{d}(x_{1,d})^2}\sqrt{\sum_{i=1}^{d}(x_{2,d})^2}}$$

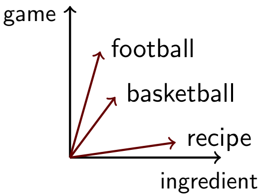
$$|\mathbf{x}| = \sqrt{\mathbf{x} \cdot \mathbf{x}} = \sqrt{x_1^2 + ... + x_d^2}$$

Vector Spaces of Text

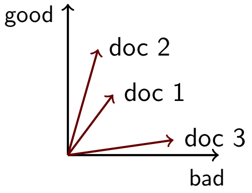
- Any ideas what are rows and columns for text data?

Vector Spaces of Text

word-word (term-context)



document-word (bag-of-words)



Why do we need vector representations of text?

- Encode the meaning of words so we can compute **semantic similarity** between them. E.g. is `basketball` more similar to `football` or `recipe`?

Why do we need vector representations of text?

- Encode the meaning of words so we can compute **semantic similarity** between them. E.g. is `basketball` more similar to `football` or `recipe`?
- **Document retrieval**, e.g. retrieve documents relevant to a query (web search)

Why do we need vector representations of text?

- Encode the meaning of words so we can compute **semantic similarity** between them. E.g. is `basketball` more similar to `football` or `recipe`?
- **Document retrieval**, e.g. retrieve documents relevant to a query (web search)
- Apply **Machine Learning** on textual data, e.g. clustering/classification algorithms operate on vectors. We are going to see a lot of this during this course!

Text units

Raw text:

As far as I'm concerned, this is Lynch at his best. 'Lost Highway' is a dark, violent, surreal, beautiful, hallucinatory masterpiece. 10 out of 10 stars.

Text units

Raw text:

As far as I'm concerned, this is Lynch at his best. 'Lost Highway' is a dark, violent, surreal, beautiful, hallucinatory masterpiece. 10 out of 10 stars.

- **word (token/term)**: a sequence of one or more characters excluding whitespaces. Sometimes it includes n-grams.

Text units

Raw text:

As far as I'm concerned, this is Lynch at his best. 'Lost Highway' is a dark, violent, surreal, beautiful, hallucinatory masterpiece. 10 out of 10 stars.

- **word (token/term)**: a sequence of one or more characters excluding whitespaces. Sometimes it includes n-grams.
- **document (text sequence/snippet)**: sentence, paragraph, section, chapter, entire document, search query, social media post, transcribed utterance, pseudo-documents (e.g. all tweets posted by a user), etc.

Text units

Raw text:

As far as I'm concerned, this is Lynch at his best. 'Lost Highway' is a dark, violent, surreal, beautiful, hallucinatory masterpiece. 10 out of 10 stars.

- **word (token/term)**: a sequence of one or more characters excluding whitespaces. Sometimes it includes n-grams.
- **document (text sequence/snippet)**: sentence, paragraph, section, chapter, entire document, search query, social media post, transcribed utterance, pseudo-documents (e.g. all tweets posted by a user), etc.
- How can we go from **raw** text to a **vector**?

Text Processing: Tokenisation

Tokenisation to obtain tokens from raw text. Simplest form: **split text on whitespaces** or use **regular expressions**.

Raw text:

As far as I'm concerned, this is Lynch at his best. 'Lost Highway' is a dark, violent, surreal, beautiful, hallucinatory masterpiece. 10 out of 10 stars.

Tokenised text:

As far as I'm concerned , this is Lynch at his best . ' Lost Highway ' is a dark , violent , surreal , beautiful , hallucinatory masterpiece . 10 out of 10 stars .

Text Processing: Other pre-processing options

- Other pre-processing steps may follow: lowercasing, punctuation/number/stop/infrequent word removal and stemming (remember COM4115/6115)

Tokenised text:

As far as I'm concerned , this is Lynch at his best . ' Lost Highway ' is a dark , violent , surreal , beautiful , hallucinatory masterpiece . 10 out of 10 stars .

Pre-processed text (lowercase, punctuation/stop word removal):

concerned lynch best lost highway dark violent surreal beautiful hallucinatory masterpiece 10 10 stars

Decide what/how do you want to represent: Obtain a vocabulary

Assume a corpus D of m pre-processed texts (e.g. a set of movie reviews or tweets).

The vocabulary $\mathcal{V}$ is a set containing all the k unique words w_i in D:

$$\mathcal{V} = \{w_1, ..., w_k\}$$

and often is extended to include n-grams (contiguous sequences of n words).

Words: Discrete vectors

Text:

love pineapple apricot apple chocolate apple pie

- $\mathcal{V} = \{$ apple, apricot, chocolate, love, pie, pineapple $\}$
- Vocabulary size: $|\mathcal{V}| = 6$

$$\text{apricot} = \mathbf{x}_2$$
$$\text{pineapple} = \mathbf{x}_3$$

Words: Discrete vectors

Text:

love pineapple apricot apple chocolate apple pie

- $\mathcal{V} = \{$ apple, apricot, chocolate, love, pie, pineapple $\}$
- Vocabulary size: $|\mathcal{V}| = 6$

$$\text{apricot} = \mathbf{x}_2 = [0, 1, 0, 0, 0, 0]$$
$$\text{pineapple} = \mathbf{x}_3 = [0, 0, 0, 0, 0, 1]$$

Words: Discrete vectors

Text:

love pineapple apricot apple chocolate apple pie

- $\mathcal{V} = \{$ apple, apricot, chocolate, love, pie, pineapple $\}$
- Vocabulary size: $|\mathcal{V}| = 6$

$$\text{apricot} = \mathbf{x}_2 = [0, 1, 0, 0, 0, 0]$$
$$\text{pineapple} = \mathbf{x}_3 = [0, 0, 0, 0, 0, 1]$$

- Also known as **one-hot encoding**. What's the problem?

Problems with discrete vectors

$$\text{apricot} = \mathbf{x}_2 = [0, 1, 0, 0, 0, 0]$$
$$\text{pineapple} = \mathbf{x}_3 = [0, 0, 0, 0, 0, 1]$$

$$\begin{aligned}
\text{dot}(\mathbf{x}_2, \mathbf{x}_3) &= 0 \cdot 0 + 1 \cdot 0 + 0 \cdot 0 + 0 \cdot 0 + 0 \cdot 0 + 0 \cdot 1 \\
&= 0 \\
\text{cosine}(\mathbf{x}_2, \mathbf{x}_3) &= \frac{\mathbf{x}_2 \cdot \mathbf{x}_3}{|\mathbf{x}_2||\mathbf{x}_3|} = \frac{0}{1 \cdot 1} = 0
\end{aligned}$$

- Every word is equally different from every other word. But *apricot* and *pineapple* are related!
- Would **contextual information** be useful?

A quick test: What is **tesguino**?

A quick test: What is **tesguino**?

Some sentences mentioning it:

- A bottle of *tesguino* is on the table.
- Everybody likes an ice cold *tesguino*.
- *Tesguino* makes you drunk.
- We make *tesguino* out of corn.

A quick test: What is **tesguino**?

Some sentences mentioning it:

- A bottle of *tesguino* is on the table.
- Everybody likes an ice cold *tesguino*.
- *Tesguino* makes you drunk.
- We make *tesguino* out of corn.

Tesguino is a beer made from corn.

Distributional Hypothesis

Firth (1957)[1]

You shall know a word by the company it keeps!

Words appearing in similar contexts are likely to have similar meanings.

[1] John R Firth (1957). "A synopsis of linguistic theory, 1930-1955". In: *Studies in linguistic analysis*.

Words: Word-Word Matrix (term-context)

- A matrix X, $n \times m$ where $n = |\mathcal{V}|$ (target words) and $m = |\mathcal{V}_c|$ (context words)

Words: Word-Word Matrix (term-context)

- A matrix X, $n \times m$ where $n = |\mathcal{V}|$ (target words) and $m = |\mathcal{V}_c|$ (context words)
- For each word x_i in $\mathcal{V}$, count how many times it co-occurs with context words x_j

Words: Word-Word Matrix (term-context)

- A matrix X, $n \times m$ where $n = |\mathcal{V}|$ (target words) and $m = |\mathcal{V}_c|$ (context words)
- For each word x_i in $\mathcal{V}$, count how many times it co-occurs with context words x_j
- Use a context window of $\pm k$ words (to the left/right of x_i)

Words: Word-Word Matrix (term-context)

- A matrix X, $n \times m$ where $n = |\mathcal{V}|$ (target words) and $m = |\mathcal{V}_c|$ (context words)
- For each word x_i in $\mathcal{V}$, count how many times it co-occurs with context words x_j
- Use a context window of $\pm k$ words (to the left/right of x_i)
- Compute frequencies over a big corpus of documents (e.g. the entire Wikipedia)!

Words: Word-Word Matrix (term-context)

- A matrix X, $n \times m$ where $n = |\mathcal{V}|$ (target words) and $m = |\mathcal{V}_c|$ (context words)
- For each word x_i in $\mathcal{V}$, count how many times it co-occurs with context words x_j
- Use a context window of $\pm k$ words (to the left/right of x_i)
- Compute frequencies over a big corpus of documents (e.g. the entire Wikipedia)!
- Usually target and context word vocabularies are the same resulting into a square matrix.

Word-Word Matrix

sugar, a sliced lemon, a tablespoonful of	**apricot**	preserve or jam, a pinch each of,
their enjoyment. Cautiously she sampled her first	**pineapple**	and another fruit whose taste she likened
well suited to programming on the digital	**computer**.	In finding the optimal R-stage policy from
for the purpose of gathering data and	**information**	necessary for the study authorized in the

	aardvark	computer	data	pinch	result	sugar	…
apricot	0	0	0	1	0	1	
pineapple	0	0	0	1	0	1	
digital	0	2	1	0	1	0	
information	0	1	6	0	4	0	

Word-Word Matrix

sugar, a sliced lemon, a tablespoonful of	**apricot**	preserve or jam, a pinch each of,
their enjoyment. Cautiously she sampled her first	**pineapple**	and another fruit whose taste she likened
well suited to programming on the digital	**computer**.	In finding the optimal R-stage policy from
for the purpose of gathering data and	**information**	necessary for the study authorized in the

	aardvark	computer	data	pinch	result	sugar	...
apricot	0	0	0	1	0	1	
pineapple	0	0	0	1	0	1	
digital	0	2	1	0	1	0	
information	0	1	6	0	4	0	

Now **apricot** and **pineapple** vectors look more **similar**!

- cosine(**apricot**, **pineapple**) $= 1$
- cosine(**apricot**, **digital**) $= 0$

Context types

- We can refine contexts using linguistic information:
  - their part-of-speech tags (**bank_V** vs. **bank_N**)
  - syntactic dependencies (**eat_dobj** vs. **eat_subj**)
  - We will see how to extract this info soon!

Documents: Document-Word Matrix (Bag-of-Words)

- A matrix X, $|D| \times |\mathcal{V}|$ where rows are documents in corpus D, and columns are vocabulary words in $\mathcal{V}$.
- For each document, count how many times words $w \in \mathcal{V}$ appear in it.

Documents: Document-Word Matrix (Bag-of-Words)

- A matrix X, $|D| \times |\mathcal{V}|$ where rows are documents in corpus D, and columns are vocabulary words in $\mathcal{V}$.
- For each document, count how many times words $w \in \mathcal{V}$ appear in it.

	bad	good	great	terrible
Doc 1	14	1	0	5
Doc 2	2	5	3	0
Doc 3	0	2	5	0

- X can also be obtained by adding all the one-hot vectors of the words in the documents and then transpose!

Problems with counts

- Frequent words (articles, pronouns, etc.) dominate contexts without being informative.

Problems with counts

- Frequent words (articles, pronouns, etc.) dominate contexts without being informative.
- Let's add the word **the** to the contexts. It often appears with most nouns:

$$\begin{aligned}
\text{vocabulary} &= [\text{aadvark}, \text{computer}, \text{data}, \text{pinch}, \text{result}, \text{sugar}, \textit{the}] \\
\text{apricot} &= \mathbf{x}_2 = [0, 0, 0, 1, 0, 1, 30] \\
\text{digital} &= \mathbf{x}_3 = [0, 2, 1, 0, 1, 0, 45] \\
\text{cosine}(\mathbf{x}_2, \mathbf{x}_3) &= \frac{30 \cdot 45}{\sqrt{902} \cdot \sqrt{2031}} = 0.997
\end{aligned}$$

Problems with counts

- Frequent words (articles, pronouns, etc.) dominate contexts without being informative.
- Let's add the word **the** to the contexts. It often appears with most nouns:

$$
\begin{aligned}
\text{vocabulary} &= [\text{aadvark}, \text{computer}, \text{data}, \text{pinch}, \text{result}, \text{sugar}, \textit{the}] \\
\text{apricot} &= \mathbf{x}_2 = [0, 0, 0, 1, 0, 1, 30] \\
\text{digital} &= \mathbf{x}_3 = [0, 2, 1, 0, 1, 0, 45] \\
\text{cosine}(\mathbf{x}_2, \mathbf{x}_3) &= \frac{30 \cdot 45}{\sqrt{902} \cdot \sqrt{2031}} = 0.997
\end{aligned}
$$

- This also holds for the document-word matrix! Solution: **Weight the vectors!**

Weighting the Word-Word Matrix: Distance discount

Weight contexts according to the distance from the word: the further away, the lower the weight!

- For a window size $\pm k$, multiply the context word at each position as $\frac{k - distance}{k}$, e.g. for $k = 3$:

$$[\frac{1}{3}, \frac{2}{3}, \frac{3}{3}, word, \frac{3}{3}, \frac{2}{3}, \frac{1}{3}]$$

Weighting the Word-Word Matrix: Pointwise Mutual Information

Pointwise Mutual Information (PMI): how often two words w_i and w_j occur togethter relative to occur independently:

$$PMI(w_i, w_j) = \log_2 \frac{P(w_i, w_j)}{P(w_i)P(w_j)} = \frac{\#(w_i, w_j)|D|}{\#(w_1) \cdot \#(w_j)}$$

$$P(w_i, w_j) = \frac{\#(w_i, w_j)}{|D|}, P(w_i) = \frac{\#(w_i)}{|D|}$$

where $\#(\cdot)$ denotes count and $|D|$ number of observed word-context word pairs in the corpus.

Weighting the Word-Word Matrix: Pointwise Mutual Information

Pointwise Mutual Information (PMI): how often two words w_i and w_j occur togethter relative to occur independently:

$$PMI(w_i, w_j) = \log_2 \frac{P(w_i, w_j)}{P(w_i)P(w_j)} = \frac{\#(w_i, w_j)|D|}{\#(w_1) \cdot \#(w_j)}$$

$$P(w_i, w_j) = \frac{\#(w_i, w_j)}{|D|}, P(w_i) = \frac{\#(w_i)}{|D|}$$

where $\#(\cdot)$ denotes count and $|D|$ number of observed word-context word pairs in the corpus.

- Positive values quantify relatedness. Use PMI instead of counts.
- Negative values? Usually ignored (positive PMI):

$$PPMI(w_i, w_j) = \max(PMI(w_i, w_j), 0)$$

Weighting the Document-Word Matrix: TF.IDF

- Penalise words appearing in many documents.
- Multiply word frequencies with their inverted document frequencies:

$$idf_w = log_{10}\frac{N}{df_w}$$

where N is the number of documents in the corpus, df_w is document frequency of word w

Weighting the Document-Word Matrix: TF.IDF

- Penalise words appearing in many documents.
- Multiply word frequencies with their inverted document frequencies:

$$idf_w = log_{10} \frac{N}{df_w}$$

  where N is the number of documents in the corpus, df_w is document frequency of word w

- To obtain:

$$x_{id} = tf_{id} log_{10} \frac{N}{df_{id}}$$

- We can also squash the raw frequency (tf), by using its log_{10}.

Problems with Dimensionality

- Count-based matrices (for words and documents) often work well, but:
  - high dimensional: vocabulary size could be millions!
  - very sparse: words co-occur only with a small number of words; documents contain only a very small subset of the vocabulary

Problems with Dimensionality

- Count-based matrices (for words and documents) often work well, but:
  - high dimensional: vocabulary size could be millions!
  - very sparse: words co-occur only with a small number of words; documents contain only a very small subset of the vocabulary
- Solution: **Dimensionality Reduction to the rescue!**

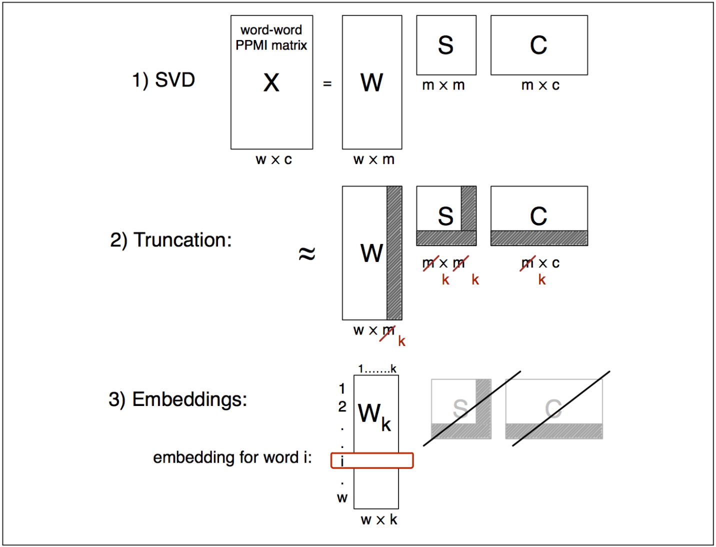
Truncated Singular Value Decomposition

- A method for finding the most important dimensions of a data set, those dimensions along which the data varies the most by decomposing the matrix into latent factors.
- Truncated Singular Value Decomposition (truncated-SVD):

$$X^{n \times m} \approx U^{n \times k} S^{k \times k} V^{k \times m}$$

- Approximation is good: exploits redundancy to remove noise by learning a low-dimensional latent space.
- For a detailed description see this tutorial.

Singular Value Decomposition on Word-Word matrix



Singular Value Decomposition on Document-Word matrix

- Also called Latent Semantic Analysis[2] (LSA)
- $U^{n \times k}$ represents document embeddings
- $V^{k \times m}$ represents word embeddings
- You can obtain an embedding $\mathbf{u}_{\text{new}}$ for a new document $\mathbf{x}_{\text{new}}$ by projecting its count vector to the latent space:

$$\mathbf{u}_{\text{new}} = \mathbf{x}_{\text{new}} v_k^\top$$

[2] Susan T Dumais et al. (1988). "Using latent semantic analysis to improve access to textual information". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 281–285.

Evaluation: Word Vectors

- Intrinsic:
  - similarity: order word pairs according to their semantic similarity
  - in-context similarity: substitute a word in a sentence without chagning its meaning.
  - analogy: Athens is to Greece what Rome is to ...?

Evaluation: Word Vectors

- Intrinsic:
  - similarity: order word pairs according to their semantic similarity
  - in-context similarity: substitute a word in a sentence without chagning its meaning.
  - analogy: Athens is to Greece what Rome is to ...?
- Extrinsic: use them to improve performance in a task, i.e. instead of bag of words → **bag of word vectors (embeddings)**

Best word vectors?

- high-dimensional count-based?
- low-dimensional with SVD? (In Lecture 8, we will see how we can obtain low-dimensional vectors with Neural Networks)
- Levy et al.[3] (2015) showed that choice of context window size, rare word removal, etc. matter more.
- Choice of texts to obtain the counts matters. More text is better, and low-dimensional methods scale better.

[3] Omer Levy, Yoav Goldberg, and Ido Dagan (2015). "Improving Distributional Similarity with Lessons Learned from Word Embeddings". In: *Transactions of the Association for Computational Linguistics*, pp. 211–225.

Limitations: Word Vectors

- **Polysemy:** All occurrences of a word (and all its senses) are represented by one vector.
  - Given a task, it is often useful to adapt the word vectors to represent the appropriate sense
- **Antonyms** appear in similar contexts, hard to distinguish them from synonyms
- **Compositionality:** what is the meaning of a sequence of words?
  - while we might be able to obtain context vectors for short phrases, this doesn't scale to whole sentences, paragraphs, etc.
  - Solution: combine word vectors, i.e. add/multiply
  - Soon we will see methods to learn embeddings for word sequences from word embeddings, the recurrent neural networks!

Evaluation: Document Vectors

- Intrinsic:
  - document similarity
  - information retrieval

Evaluation: Document Vectors

- Intrinsic:
  - document similarity
  - information retrieval
- Extrinsic:
  - text classification, plagiarism detection etc.

Limitations: Document Vectors

- Word order is ignored, but language is sequential!

Upcoming next...

- Document vectors $+$ machine learning $\rightarrow$ text classification!

Reading Material

- 6-1 to 6-7, 6-9 to 6-12 from [Chapter 6] Jurafsky & Martin
- Vector space models of semantics [paper]